Dependability-Driven Runtime Management of Service Oriented Architectures

Haouas Hanen, Johann Bourcier

▶ **To cite this version:**

Haouas Hanen, Johann Bourcier. Dependability-Driven Runtime Management of Service Oriented Architectures. PESOS - 4th International Workshop on Principles of Engineering Service-Oriented Systems - 2012, Jun 2012, Zurich, Switzerland. hal-00714357

HAL Id: hal-00714357
https://inria.hal.science/hal-00714357

Submitted on 4 Jul 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Dependability-Driven Runtime Management of Service Oriented Architectures

Hanen Haouas
INRIA
Campus de Beaulieu
35042 Rennes Cedex, FRANCE
hanen.haouas@inria.fr

Johann Bourcier
IRISA, Universite de Rennes1, INRIA
Campus de Beaulieu
35042 Rennes Cedex, FRANCE
johann.bourcier@irisa.fr

***Abstract*—Software systems are becoming more and more complex due to the integration of large scale distributed entities and the continuous evolution of these new infrastructures. All these systems are progressively integrated in our daily environment and their increasing importance have raised a dependability issue. While Service oriented architecture is providing a good level of abstraction to deal with the complexity and heterogeneity of these new infrastructures, current approaches are limited in their ability to monitor and ensure the system dependability. In this paper, we propose a framework for the autonomic management of service oriented application based on a dependability objective. Our framework proposes a novel approach which leverages peer to peer evaluation of service providers to assess the system dependability. Based on this evaluation, we propose various strategies to dynamically adapt the system to maintain the dependability level of the system to the desired objective.**

***Keywords:* Services, Dependability, Autonomic Computing, EnTiMid, Kevoree.**

I. INTRODUCTION

The emerging complexity in distributed systems, services and applications has raised unprecedented challenges for system management. With the aim of building large scale systems while reducing cost and dealing with complexity, dynamism, heterogeneity and uncertainty, autonomic computing principles have paved necessary foundations towards self-managing systems that are self-configuring, self-healing, self-optimizing, and self-protecting [1]. This new paradigm has been adapted into Service Oriented Architecture (SOA) [2, 3]: an architecture offering services to applications or other services through published and discoverable interfaces.

Supported by this architecture, services have become the basic blocks for building information systems from loosely-coupled elements. The binding of services into flexible compositions is recognized as a powerful paradigm for building distributed applications. However, the Quality of Service (QoS) delivered by these systems remains an important concern, and needs to be managed in an equally adaptive and predictable way.

The IFIP Working Group on Dependable Computing and Fault Tolerance [16] makes the following statement on dependability. "The notion of dependability, defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers, enables these various concerns to be subsumed within a single conceptual framework."

Current techniques to build Service Oriented application are limited in their ability to observe and ensure the dependability of these applications especially when this dependability is meant to evolve during the application life. This paper tackles the problem of dynamic service reconfiguration based on a dependability objective provided by an administrator.

Current quality management designs for service-oriented systems are inadequate for ensuring runtime system quality of service as they focus on static service properties, rather than emergent properties. Most solutions provided in this field are very problem-specific and do not suit our particular needs.

This paper describes a generic consumer-centered runtime architecture that combines service monitoring, reputation, evaluation, decision making and reconfiguration to provide a self-managing mechanism for ensuring runtime quality in service-oriented systems. Our approach represents a closed-loop control system that first monitors and evaluates the system, then analyzes and takes decisions and finally provides system reconfiguration execution.

The rest of the paper is structured as follows. In section 2, we introduce the motivation and the scientific context of our work. In section 3 an overview of our approach is given in detail. Section 4 presents some related work and in section 5 we present the scenario of application that we have developed. Our solution is then evaluated in section 6.

II. MOTIVATION AND PROBLEM STATEMENT

The elderly population is growing and there is a need to care for increasing numbers with less money [7]. Therefore, home automation is being implemented into more and more homes of the elderly [6, 8]. These systems are built based on the idea that business function is provided as a series of services, which are assembled together to create solutions that serve a particular business need. Unfortunately, building such

applications remains very complex and raises various scientific challenges especially that there is an emerging tendency to build these systems at runtime. One of the most challenging is the capacity of these applications to dynamically adapt at runtime to their execution environment. In such dynamic environment, the dependability of applications is very hard to ensure, while being a crucial requirement for end users. Even systems that are usually safe and dependable can fail as the result of complex interactions between their different software services [10].

In this context, we can mention ENTIMID [11]: a runtime environment built on top of a service-based architecture to support evolutions, adaptations and openness required by the proposed model. In this paper we consider this runtime representation to propose a new approach involving autonomic managers that are capable of considering the architecture of a service-based application to continuously adapt smart building applications dependability. To this end, we first evaluate the dependability of an application and the whole system and then use these data to dynamically optimize the software dependability.

III. OVERVIEW OF OUR APPROACH

In the field of systems engineering, dependability is defined as the "Quality of the delivered service such that reliance can be justifiably placed on this service." The main focus of this paper, concerns the dynamic adaptation of a SOA system to obtain a targeted dependability. Thus our main proposition follows classical closed-loop system architectures, by first monitoring and evaluating the system, then analyzing and taking decisions regarding the adaptations of the system and finally providing system reconfiguration execution. As illustrated in Figure 1, our entire self-adaptive behavior is driven by a targeted dependability for the managed system.

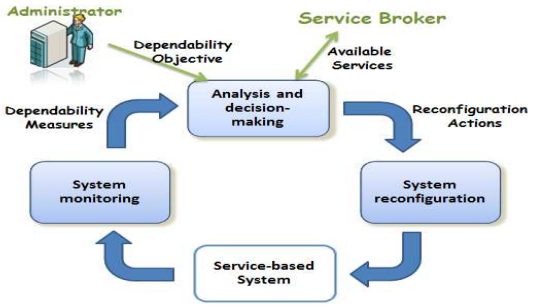

Figure 1. Our approach

We have adopted a control systems perspective: a system is monitored and the resulting evaluations are used to determine system health based on an objective dependability value, and then the system is adapted to fix existing problems and achieve the desired outcomes.

A service provider and a service consumer interact according to a negotiated contract named Service Level Agreement (SLA) describing the specifications of the quality characteristics that should be met by the provider. When a service provider is not fulfilling the requirements described in the SLA, the dependability of this provider is negatively impacted. In the remaining of this section, we describe the dependability evaluation of a service provider and the impact of this evaluation on the whole system dependability.

A. *Dependability Evaluation Based On System Monitoring*

We suggest a new representation of the dependability by gathering the dependability attributes and combining them with the concepts of trust and reputation. The basic idea is to let parties rate each other. The aggregated ratings about a given service component will be used then to derive a trust or reputation score, which can assist other parties in deciding whether or not to select that component in the future. The propagation and the aggregation of the different dependability values combined with the consumers' appreciations enable the evaluation of the whole system dependability.

1) Calculating the Dependability of an Application: To estimate the dependability of an application, we provide a mechanism which involves the clients' participation to gather run-time information related to the services, to share information with the services about the interactions and to evaluate the QoS of the Web Services.

a) *Voting based Approach:* Assuming we have two bound service components; a service provider and a service consumer. The consumer evaluates the quality of the service consumed and compares it to the agreed service levels. This comparison is used to derive a score reflecting his appreciation of each contracted QoS attribute.

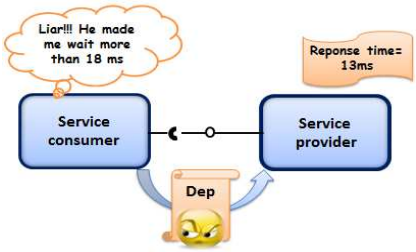

Figure 2. Dependability voting approach

Each client bound to a service provider, will apply these steps to evaluate every QoS parameter and then aggregates the resulting values by using a weighted average. The aggregated result represents the appreciation of the consumer and the new supplier's dependability. This method enables the evaluation of the dependability of each software component.

2) Calculating the Dependability of the System: To continuously determine the reputation of the whole system we gather and exploit the dependability of each service

provider to provide a global dependability metric for the system.

a) *Dependability Propagation Approach:* The example illustrated above can be generalized to have multiple consumers and providers. In this case, a peer 2 peer voting process is initiated. Each supplier will have a new dependability value computed according to the evaluation of his service by the concerned consumers. A software service having low dependability value has bad reputation and has probably failed. Therefore, we cannot totally rely on the evaluation it performs. Thus, we developed a trust mechanism based on the reputation concept. Our mechanism builds trust through a dependability propagation approach, as illustrated in Figure 3.

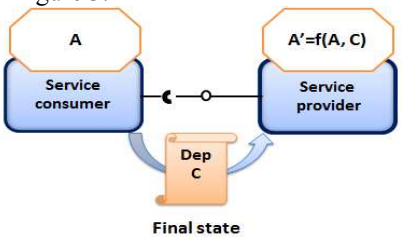

Figure 3. Dependability propagation approach

b) *Dependability Aggregation Approach:* Let's now assume we have multiple consumers that have chosen the same provider. In this case, we will not have just one evaluation of the service delivered by this provider. Each consumer will rate either favorably or against this supplier according to the voting process explained above. These various evaluations have to be considered for obtaining a reliable decision in the service selection process. The determination of the reputation of the service provider necessitates the gathering of the different evaluations performed by all its customers. For this, an aggregation algorithm is used as shown in Figure 4.

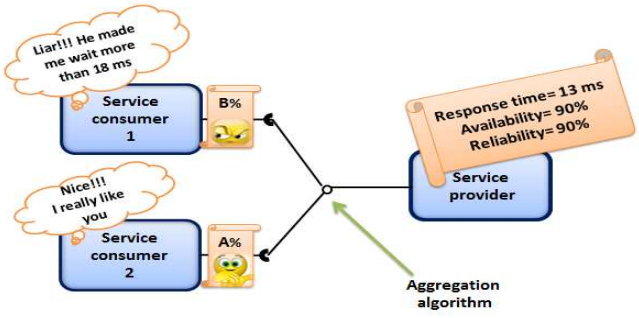

Figure 4. Dependability aggregation approach

c) Selection-based Approach: In the dependability propagation approach, the reputation of the consumer is considered while addressing the dependability of the subsystem; all dependencies between its different parts have been eliminated. Therefore, this consumer won't be taken into consideration while computing the overall system dependability. Consider a system composed of multiple interacting service components performing different tasks; if we apply our propagation approach to its parties, we will obtain a set of independent subsystems each having a dependability value reflecting the behavior of all its parties (see Figures 5)

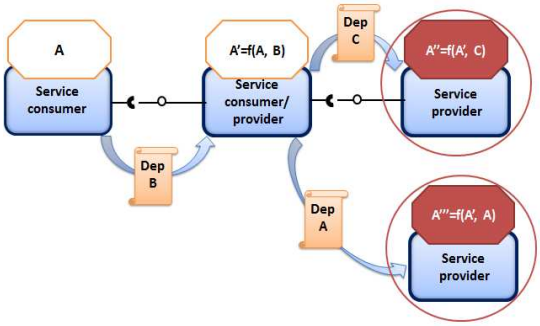

Figure 5. Dependability selection approach

d) Harmonic Mean: The different approaches detailed above aimed at simplifying the dependencies between the different entities. Now, we need to find the appropriate way to use the resulting ratings to evaluate our system. Such solution should fulfill the following criteria i) supporting independent values, ii) insuring that lower values have a greater impact than higher values since safety is crucial to achieving in smart building especially when dealing with the issue of falls in the elderly. For example, a fall detection sensor is an important service component in our case. This component may fail and thus its dependability value will decrease. In this context, the end to end dependability value should not be high even though the other components are performing properly.

In the present work, we have chosen to use the harmonic mean that is appropriate for situations when the average of rates is desired. To give a reason for choosing this method, we review the concept of harmonic average detailed in [13, 14] and search for the different applications of this method; (Baeza-Yates and Ribeiro-Neto,1999) [15]. We believe that in our situation, the harmonic mean is the best average measure that we can use to aggregate the votes while meeting the requirement of our platform.

e) *Exponential Moving Average:* The reconfiguration of the system is based on the results of dependability evaluation. To this end, we need stability in the system while still having a value reflecting the current state of the system. Exponential smoothing [11] is a technique that can be applied to time series data, either to produce smoothed data for presentation, or to make forecasts. This method emphasized the more recent values. We use this technique to represent the current situation of a supplier. Unlike some other smoothing methods, this technique does not require

any minimum number of observations to be made before it begins to produce results.

3) Architecture: In the view we are taking here, we focus on designing a software structure that will assist the management of services at runtime. We propose an architecture that captures all QoS aspects of a runtime service-based system to assess its dependability. The proposed architecture is depicted in Figure below. It consists of four major layers: a component type layer, a SLA monitoring layer, a Static dependability monitoring layer and a dynamic dependability monitoring layer.

The first layer is composed by software services. Any request to the API should go through this layer.

The second layer of the architecture is responsible for providing proper context related services that permits us to collect the evaluations of different consumers involved in the application from the component type layer and derive a reputation score based on the dependability of the sender and its service appreciation. This layer consists of two generic components: the asynchronous SLA manager responsible for the evaluation of providers in case of asynchronous communications and the synchronous SLA manager related to synchronous communications. The calculated values will be sent to the next layer to be treated.

The third layer consists of a static dependability monitor that receives the ratings from the second layer and calculates the static dependability of the system using the harmonic mean described earlier. This dependability will be communicated to the dynamic dependability monitoring layer.

Finally, the fourth layer is composed of a dynamic dependability manager that exploits the incoming data to continuously assess the system dependability by applying the moving exponential average explained above.

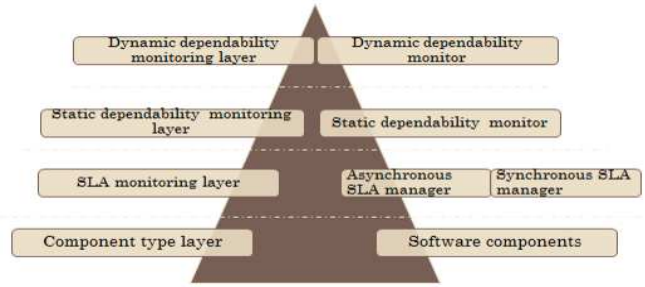

Figure 6. Four layered architecture design process

Both dynamic dependability monitor and the static dependability monitor are generic components. Each defined autonomic manager handles a derby database in which he stores the needed information to properly perform its function.

B. Analysis and Decision Making

The evaluation described in the previous section will guide us through the decision making process. We aim at ensuring that our system will satisfy the proposed SLA requirement. For this we compare the resulting evaluation with low level threshold that indicates whether overall system goals are met or not. The threshold presents our Service Level Objective (SLO) [17] and it can be determined by system administrators and experts using past experience with specific applications. This process is initiated when the dependability of the system become lower than the threshold. It consists on identifying the failed sub-system(s) and trying to find the appropriate transformation that can be applied to optimize the dependability and reconfigure the system. We mean by reconfiguration readjusting the internal structure of the system without changing its main function. Service is the target of reconfiguration. We implemented a reconfiguration manager that analyses and specifies appropriate actions to take under various situations according to the dependability of each failed sub-system, the threshold and the several available services. Our reconfiguration manager is responsible for capturing the current runtime configuration, analyzing the dependability resulting evaluation based on the service objective, querying the service broker to know about available services, locating the appropriate software services and finding all or the most suitable configuration that optimizes the system performance. It manages heterogeneous services that are published to the broker.

Various reconfiguration techniques can be applied, under different running modes. Based on different demands of services, we design the following three service reconfiguration modes.

1) Software Service Addition: Component addition means to add a new software service and reconnect components with each other according to the new component added.

2) Connection Change not Accompanied by Component Change: Connection changes means that connection change is only performed according to requirements when multiple connections are available for changing the reconfiguration. For example, the service provider 1 delivers a service that meets the expectation of service consumer 1 but doesn't satisfy the consumer 2. Thus the dependability of the subsystem 2 is lower and affects the overall system dependability. We can change the connection and reconnect consumer 2 with provider 2 to have better quality of service.

3) Feature Deletion: Feature deletion means to delete the unnecessary feature and reconnect features with each other as a result of requirement changes, etc. In load balancing against multiple features for performance enhancement, features may be reduced depending on the load.

In the example illustrated above, the subsystems composed of (Service consumer 1, Service provider 1) and (Service consumer 2, service provider 1) have low values of dependability because of the unacceptable quality of service delivered by the provider. To attain our objective, the service provider 1 has to be replaced by a new component: Service provider 3.

4) Restart Mode: The reconfiguration manager monitors the services, and restarts the service as soon as it finds abnormal service termination. In case there is no appropriate reconfiguration method found, this manager will restart the platform.

The reconfiguration manager searches the vast and complex space of possible reconfigurations with the goal of evolving suitable reconfiguration plans in response to changing requirements and environmental conditions. Depending on the scenario, we found several different reconfiguration plans. We had the choice between executing the first adequate solution found, and selecting the optimal one. We opted for the second choice because we thought that factors such as availability, response time and other QoS attributes need to be taken into consideration. The question is: how can we be sure we make the best decision, while taking all of these different factors into account?

We used Grid Analysis which is a useful technique for taking decisions in presence of various good alternatives, and many different factors to take into account. To deal with the problem of factors' importance, we weighted the rankings differently using Paired Comparison Analysis. This tool helped us to set priorities where there are conflicting possibilities such as considering that response time might be twice more important to us than cost.

C. *Executing Reconfiguration*

As mentioned in the motivation part, we consider the runtime representation of service-based architecture provided by the component based framework EnTiMid which is itself based on Kevoree. In our work we used Kevoree to execute reconfiguration actions. In fact, Kevoree is a Software Engineering tool, to efficiently build adaptable and distributed component-based applications. The platform uses models at runtime to control and coordinate the adaptation of distributed applications. Kevoree helped us to apply reconfiguration plans by adding, deleting and reconnecting software services thanks to its adaptation capabilities:

- **Parametric adaptation**: Dynamic update of parameters values, e.g. sampling rate of component wrapping a physical sensor.
- **Architectural adaptation**: Dynamic addition/removal of Bindings/components, e.g. replication of software components and bindings on different nodes for load balancing.
- **Dynamic provisioning of types**: Hot deployment of component types.
- **Remote management**: Nodes can also host and participate in a remote management layer which supervises less powerful nodes.

IV. RELATED WORK

The main objective of dependability evaluation is to assess the ability of a system to correctly function over time. There are many approaches used to evaluate the dependability of complex systems. Most of these approaches are based on probabilistic models and are mainly based on software-fault injection technique. The methods for qualitative and quantitative assessments are specific: Failure Mode and Effects Analysis (FMEA) [21] for the qualitative evaluation, and Markov chains [25], stochastic Petri networks [26] for quantitative assessment. The Reliability Block Diagrams (RBD) [22] and fault trees [23, 24] can be used to perform both types of evaluation.

In all these techniques, system dependability is measured through its attributes, such as reliability, availability, confidentiality, and integrity. Most of these measurements have been performed regardless the complex and dynamic evolution of system specification and design at run-time. These methods are usually centralized, static and applied during the test phases or before execution. In addition, it is difficult to manufacture and maintain these models in the case of large systems with many service providers [27, 28].

The dependency between several software services brings the need for trust between these parties. To this end, we have developed a trust mechanism based on the reputation concept. Our mechanism builds trust through a dependability propagation approach; a notion that can be found in similar work [31] dealing with calculating trust values based on recommendations.

Dynamic reconfiguration can be observed in different domain such us cloud computing and Wireless Sensor Networks. In [18] the conception of dynamic reconfiguration is to operate dynamic reconfiguration on cloud computing virtual services, which refers to making a recombination of the nodes topology structure, correcting service failures itself, updating existing services and system modules online, increasing and deploying new services dynamically and so on without changing the main function of cloud computing virtual service. The paper [19] introduces the node architecture that is dynamically reconfigurable to support non-functional requirements based on four elements: feature addition, deletion, moving, and connection change.

Various reconfiguration mechanisms [30] have been proposed in the field of sensor networks making a trade-off in flexibility vs. update cost. These mechanisms are listed below:

- **Full image binary upgrades**: in TinyOS provide maximum flexibility by allowing arbitrary changes to the functionality, but incur unacceptable update cost.
- **Modular binary upgrades**: in systems like SOS provide almost similar flexibility as the full image upgrade but at a significantly lower cost.
- **Virtual machines**: provide a more cost efficient way to update application level functionality of the system. Maté [29] was the first virtual machine architecture proposed for the resource constrained sensor devices.
- **Parameter Updates and Query Frameworks**: SNMS is a framework for updating parameters of TinyOS components written in NesC programming language. It has very limited flexibility but it also has a very low update cost. TinyDB is a SQL-like query framework for gathering data from sensor networks. The TinyDB framework allows re-tasking of the software by moving around points of data aggregation in the network.

The paper [20], presents a system that supports software reconfiguration in embedded sensor networks at multiple levels. The system architecture is based on an operating system consisting of a fixed tiny static kernel and binary modules that can be dynamically inserted, updated or removed. This system integrates the three design alternatives (excluding full binary upgrades) into one complete system.

Achieving high dependability of SOA is crucial for a number of emerging and existing critical domains. Most techniques presented below and used to adapt system dependability at runtime, have limitations and does not provide the level of dependability evaluation needed for complex systems especially those based on service approach and used in the field of smart houses. Our approach follows classical closed-loop system architectures, by first monitoring and evaluating the system, then analyzing and taking decisions regarding the adaptations of the system and finally providing system reconfiguration execution.

V. APPLICATION SCENARIO

The contribution of this work is to provide a software infrastructure that enables the autonomic management of service oriented application based on a dependability objective. So, to validate our approach, we have developed and tested our work based on a scenario of application based on services, adapted to smart buildings and used to prevent falls and control the health of the resident mainly an elderly person. This application is achieved using our platform prototype presented in the contribution. Our scenario is a simple application consisting of the following service components: 1) A heart rate sensor: A personal monitoring device providing information on the heart rate. 2) A health manager: A component which verify the heart rate and raises alert. 3) An alert manager which manages alert requests. 4) A mailing manager which sends alert via email to a predefined nurse or emergency unit 5) A synchronous / asynchronous SLA manager which receives data and computes the supplier's dependability. 6) A Static dependability manager which stores dependability values of all software components, and computes the dependability of the system. 7) A dynamic dependability manager, which stores past values of system dependability.

In case of non respect of the giving dependability objective, the reconfiguration manager searches the space of possible reconfigurations (see section 3). The Figure below illustrates the scenario and the architecture of the application.

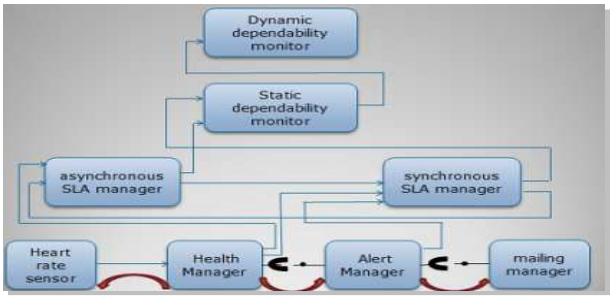

Figure 7. Example of Component Based Services Application in the field of smart building

VI. VALIDATION

We have tested the efficiency, the accuracy and the distribution of our approach on various configurations. For example, we have simulated the presence of a elderly in the home. During these test we have simulated various scenario such as sudden change in the dependability value of component or changing the heart rate of the elderly.

We found that with our distributed architecture, the services are properly evaluated according to consumers' appreciations and their reputations. The derived scores are aggregated and these values are used to adapt the dependability of the system through different reconfiguration plans. A low dependability value of a component has a deep impact on its reputation.

For instance, we use the heart rate sensor to prevent falls or to help react faster in case of falls. This device may fail and thus providing inaccurate data. This failure may lead to the death of this person. Our application, correctly responds to such problem and the resulting end to end dependability value traduces the behavior of the system and its responsiveness to rapidly changing conditions.

Our approach is based on generic autonomic managers. Our method does not affect the response time of the system since we just intercept messages. Motivated by the problems of home automation for the elderly, our proposed framework is not specific to the problems of home automation; it is generic and intended for all service compositions. It should be emphasized that calculation methods for assessing the dependability depends on the context and can be further

extended to encompass more expressive and complicated mechanisms adapted to other types of systems and enhanced to incorporate other QoS attributes.

VII. CONCLUSION

In this work, we were interested in studying dynamic service reconfiguration based on a dependability objective and related to pervasive applications in the context of home automation for healthcare. However, as software systems become ubiquitous, the capacity of these applications to dynamically adapt at runtime to their execution environment is being questioned and the issues of dependability become more and more important and hard to ensure, while being a crucial requirement for end users.

In this paper we introduced a novel service-based approach for ensuring runtime quality in service-oriented systems. This consists in an infrastructure which continuously evaluates the dependability, analyses and takes decision regarding the adaptations of the system and finally providing system reconfiguration execution. Our framework provides also means to extend the evaluation methods to improve the dependability representation. This work was carried out based on the component platform KEVOREE.

As future work, we are interested in studying dependability in the context of a Process Industry Engineering. Our proposed approach will be tested and applied in the field of Process Industry Engineering.

REFERENCES

[1] Salim H., Bithika K., Houping C., Jingmei Y., Yeliang Z., Manish P. and Hua L., Computing :The Autonomic Computing Paradigm, Vol. 9, n°1, p. 5-17, January 2006.

[2] SSA Research Note SPA-401-068, 'Service Oriented Architectures, Part 1', 12 April 1996.

[3] SSA Research Note SPA-401-069, 'Service Oriented Architectures, Part 2', 12 April 1996.

[4] Marc Oriol H. Quality of Service (QoS) in SOA Systems. p.77-78. October 2009.

[5] IFIP WG 10.4 - Dependable Computing and Fault Tolerance : 'Dependability :Basic Concepts and Terminology', p.5-6, 1994.

[6] Richard W. Sattin. Falls among older persons : A public health Perspective, Vol.13, p. 439-440, 1992.

[7] Vellas BJ, Wayne SJ, Romero LJ, Baumgartner RN, Garry PJ. Fear of falling and restriction of mobility in elderly fallers. Age and Ageing, vol.26, p. 189-193, 1997.

[8] Tinetti ME, Speechley M. Prevention of falls among the elderly. New England Journal of Medicine 1989 ; 320(16) :1055-9.

[9] M. P. Papazoglou, 'Service-oriented computing : Concepts, characteristics and directions,' in Fourth International Conference on Web Information Systems Engineering (WISE'03), 2003;

[10] Grégory N., Olivier B. Régis F. and Jean-Marc J., 'EnTiMid : un middleware au service de la maison', 2009

[11] Adam M., Simple, 'Exponential, and Weighted Moving Averages', http://daytrading.about.com/od/indicators/a/MovingAverages.htm, Mai 02, 2011.

[12] Hanzak, T. irregular-periodic-time-series (Sep 8, 2008), 1-33.

[13] Bin Z., Meichun H., and Umeshwar D., 'K-Harmonic Means - A Data Clustering Algorithm', p. 1-4,1999.

[14] Mazen O. H., Mohamed-Slim A., IEEE TRANSACTIONS ON COMMUNICATIONS. Harmonic Mean and End-to-End Performance of Transmission Systems With Relays, JANUARY 2004, VOL. 52, NO. 1, p.130-131.

[15] Wei S., Sumit R., Conference : Global Telecommunications Conference - GLOBECOM: Achieving Full Diversity by Selection in Arbitrary Multi-Hop Amplifyand-Forward Relay Networks, p. 1-6, 2009.

[16] IFIP WG10.4 on Dependable Computing and Fault Tolerance.

[17] http://www.knowledgetransfer.net/dictionary/ITIL/en/Service_ Level_Objective.htm.

[18] Shuai Wang 1, Fang Du 2, Xinming Li 1, Yi Li 1, Xingye Han1, Proceedings of IEEE CCIS2011 "Research on dynamic reconfiguration technology of cloud computing virtual services"

[19] HAYASHI Takeo, UENO Hiroshi, R&D supporting future cloud computing infrastructure technologies; "Dynamically Reconfigurable Network Nodes in Cloud Computing Systems", IT/Network Integrated Control Plane

[20] Rahul Balani, ChihChieh Han, Ram Kumar Rengaswamy, Ilias Tsigkogiannis; "Multilevel Software Reconfiguration for Sensor Networks", Published by ACM 2006 Article.

[21] Hasan S., Bedir T. and Mehmet A., Architecting dependable systems IV : Extending failure modes and e_ects analysis approach for reliability analysis at the software architecture design level, 2007.

[22] Modarres, Mohammad; Mark K. and Vasiliy K. Reliability Engineering and Risk Analysis. Ney York, NY : Marcel Decker, p. 198. ISBN 0-8247-2000-8. http ://books.google.co.uk/, April 2011.

[23] Texas Department of Insurance, 'Fault Tree Analysis', HS02-015B (9-06).

[24] A. Bobbio a, L. Portinale a, M. Minichino b and E. Ciancamerla b, 'Improving the analysis of dependable systems by mapping fault trees into Bayesian networks', p.249-260, 2001.

[25] Ricardo M. Fricks and Kishor S. Trivedi, 'Importance Analysis with Markov Chains', 2003RM-102 : p. 1-7.

[26] Y.Dutuita, E.Châteletb, J.-P.Signoretc and P.Thomasa, 'Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases', May 1998.

[27] R. de Lemos et al. (Eds.), Architecting Dependable Systems IV: 'Extending Failure Modes and E_ects Analysis Approach for Reliability Analysis at the Software ArchitectureDesign Level', LNCS 4615, p. 409433, 2007.

[28] Salvatore D., and Messina A., 'Dependability Evaluation with Dynamic Reliability Block Diagrams and Dynamic Fault Trees', Vol. 6 Issue 1, January 2009

[29] Philip L. and David C. "Maté: A Tiny Virtual Machine for Sensor Networks." In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X), 2002.

[30] R. Balani, C. C. Han, et al., "Multi-level software reconfiguration for sensor networks," In EMSOFT'06.

[31] Hang, C., Singh, M.: Trust-Based Recommendation Based on Graph Similarity. In: Proceedings of the 13th AAMAS Workshop on Trust in Agent Societi